Name:

ID:

CSCI 3104, Algorithms **Profs. Chen & Grochow**
Problem Set 1 – Due Jan 24 11:55pm **Spring 2020, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

Advice 2: Informal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Instructions for submitting your solutions:

- The solutions **should be typed** and we cannot accept hand-written solutions. Here's a short intro to LaTeX.
- You should submit your work through the **class Canvas page** only.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this template of at least 9 pages (or Gradescope has issues with it).

Quicklinks: 1 2a 2b 2c 3 4a 4b 4c 4d

---

1. *What are the three components of a loop invariant proof? Write a 1–2-sentence description for each one.*

   Initialization: The loop invariant is true prior to the first iteration of the loop.
   Maintenance: If the loop invariant is true before iteration i of the loop, it remains true before iteration $i + 1$.
   Termination: When the loop terminates, the invariant gives us a useful property that helps show that the algorithm is correct.

Name: _____

ID: _____

CSCI 3104, Algorithms **Profs. Chen & Grochow**

Problem Set 1 – Due Jan 24 11:55pm **Spring 2020, CU-Boulder**

---

2. *Identify and state a useful loop invariant in the following algorithms. You* do not *need to prove anything about it.*

   (a)
   ```
   FindMinElement(A) : //array A is not empty
   ret = A[length(A)]
   for i = 1 to length(A)-1 {
       if A[length(A)-i] < ret{
           ret = A[length(A)-i]
   }}
   return ret
   ```

   Prior to the start of iteration i:

   `ret` $= \min\{A[n], A[n-1], \ldots, A[n-i+1]\}$

(b)
```
LinearSearch(A, v) : //array A is not empty and has no duplicates
    ret = -1 //index -1 implies the element haven't been found yet
    for i = 1 to length(A) {
        if A[i] == v{
            ret = i
    }}
    return ret
```

Prior to the start of iteration i:

If the query element v is in $A[1..i-1]$ at position k, then `ret` $= k$, and otherwise `ret` $= -1$